XRootD at RAL

Jyothish Thomas, James Walder, Thomas Byrne
jyothish.thomas@stfc.ac.uk, james.walder@stfc.ac.uk, tom.byrne@stfc.ac.uk

XrootD

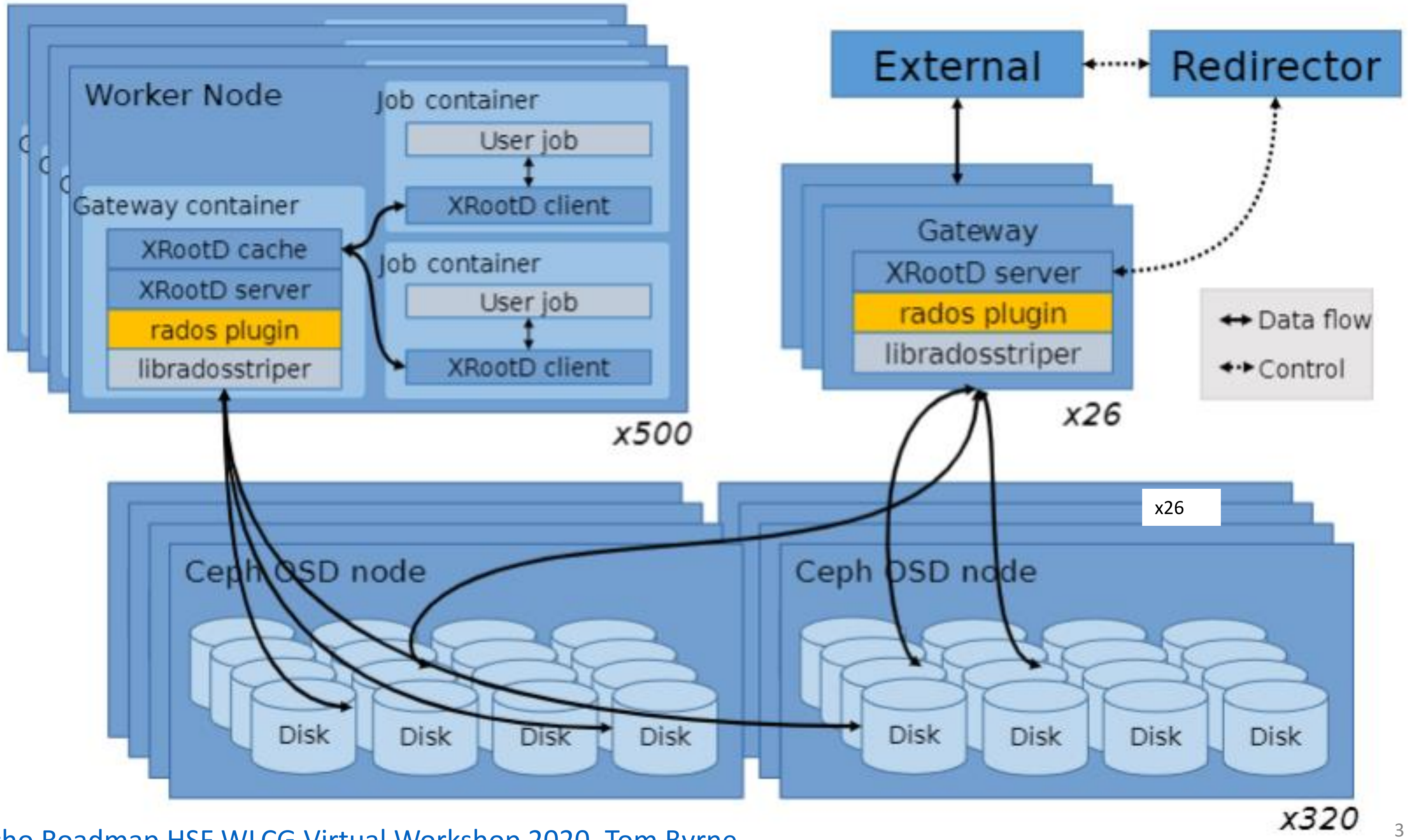
Overview

- The XROOTD project aims at giving access to data repositories of many kinds.
- It is based on:
    - a scalable architecture (client/server services)
    - a communication protocol (root)
    - and a set of plugins and tools based on those. (e.g. XrdCeph)
- It provides features such as authentication/authorization, integrations with other systems, etc..

UK collaboration meetings

- Wednesdays GridPP Storage Meeting – Gereral storage meeting
- Thursdays XrootD Meeting – xrootd specific meeting

[1] https://xrootd.slac.stanford.edu/
[2] https://stfc.atlassian.net/wiki/spaces/X/overview



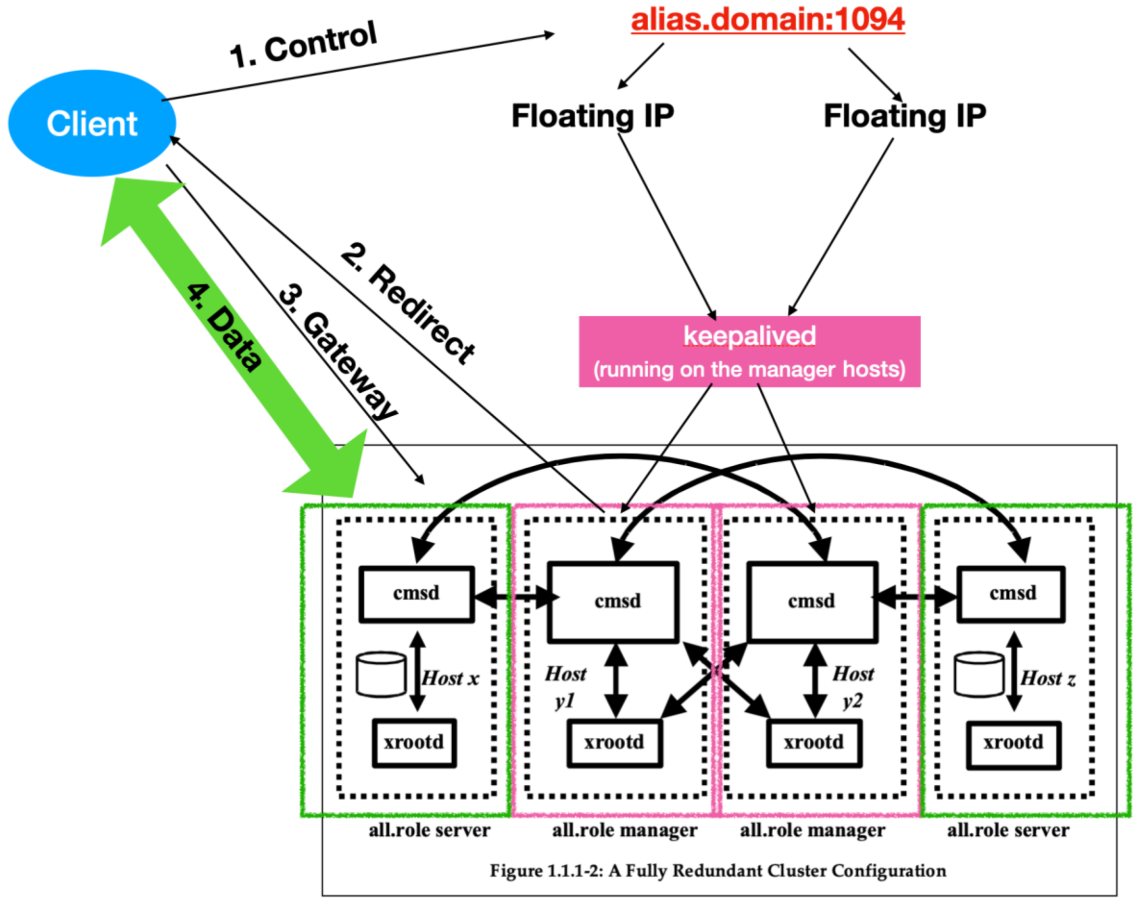
[1] Echo Roadmap HSF WLCG Virtual Workshop 2020, Tom Byrne

Current Status

- Redirectors…
  - XrootD Cluster Management Service (CMSD) redirectors are in production and used by all VOs for webdav and root transfers through the external gateways (webdav.echo.stfc.ac.uk, xrootd.echo.stfc.ac.uk)
- Additional Gateways
  - Currently 26 gateways in production use (21 common, 3 alice, 2 cms-aaa)
- Tokens are enabled
- Network tuning
- Checksum speed improved (more on LHCb update by Alexander Rogovskiy)

CMSD setup

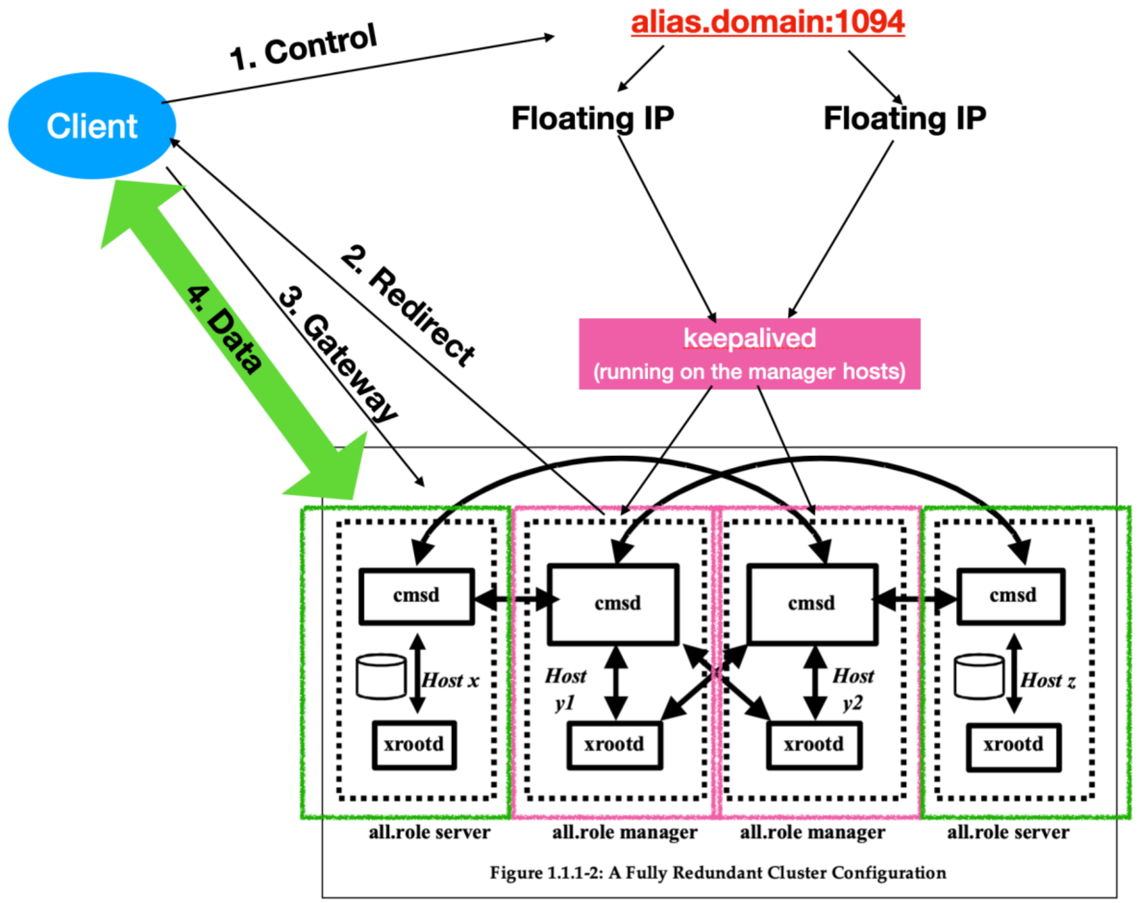
DNS ROUND ROBIN	CMSD
• If a gateway fails then it remains in the alias until manually removed • Clients can bypass the round-robin by caching a particular gateway host. No active load-balancing is possible.	• Seamlessly deal with a failed gateway or intervene on individual gateways. • Evenly spread the load between the gateway hosts and automatically mitigate the pattern of 'hotspotting' • Reduce our dependence on the DNS provider. • Allow us to use a much longer TTL for our Echo alias, and so make Echo more resilient against any DNS issues



Initial CMSD setup diagram, by James Walder and Tom Byrne

CMSD setup

DNS ROUND ROBIN	CMSD
• If a gateway fails then it remains in the alias until manually removed • Clients can bypass the round-robin by caching a particular gateway host. No active load-balancing is possible.	• Seamlessly deal with a failed gateway or intervene on individual gateways. • Evenly spread the load between the gateway hosts and automatically mitigate the pattern of 'hotspotting' • Reduce our dependence on the DNS provider. • Allow us to use a much longer TTL for our Echo alias, and so make Echo more resilient against any DNS issues



Initial CMSD setup diagram, by James Walder and Tom Byrne

The XrootD load balancing algorithm

How it’s intended to work:

1. Generate an overall load score based on a weighted sum of the different load metrics reported (network, cpu load, system load, memory usage, disk space)
2. Skip unusable nodes (not responding, over the configured max load, etc..)
3. Assign incoming transfers by round robinin between the least loaded gateway and other gateways within a set window (fuzz) around it

The XrootD load balancing algorithm

How it works[1]:

1. Generate an overall load score based on a weighted sum of the different load metrics reported (network, cpu load, system load, memory usage, disk space)
2. Skip unusable nodes (not responding, over the configured max load, etc..)
3. go through the gateways in order of first appearance in the cluster, switching the selected gateway to the next one if it’s significantly less loaded or within the fuzz and had received less transfers than the currently selected one

[1] XRootD CMSD SelByLoad Analysis, Thomas Byrne 2024

The XrootD load balancing algorithm

- The transfers will always hit the best N gateways
- Can lead to bouncing between different subset of gateways selected by the load balancer
- Some load patterns result in problematic behaviours (see appendix)

The 5 phases of XrootD load balancing

1. Bargain
2. Explore
3. Nostalgia
4. Vintage
5. Innovate

Phase 1 – Bargain

Tune the existing load balancing to distribute load very evenly

- 80/20 split of system load/cpu, with fuzz 3 provided the best load balancing we could get
- Generally ok but performance degrades under heavy load

Phase 2 – Exploration

Explore the space for better alternatives under heavy load

- 50/50 split of system network/cpu
  - no significant difference. Some improvement in performance for newer hardware at the expense of the older ones
- Non-standard metrics
  - Number of active connections, heartbeat time
  - Not very consistent and hard to tune equally among gateways under heavy load

Phase 3 - Nostalgia

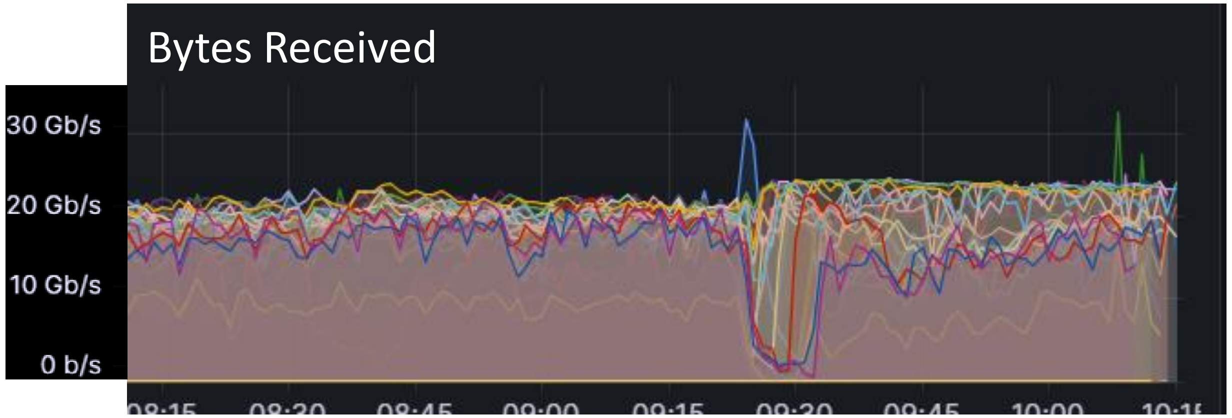
Simulate Round Robin

- All gateways report the same loads artificially (passive load balancing)
- A lot more stable, low error rate and better throughput even under heavy load
- If an individual gateway starts to get loaded, it will keep getting loaded until it breaks

Phase 4 – Vintage

Gateways report the same load unless it’s nearing problematic levels (80% system load) at which point the reported load is set higher to remove it from the Round Robin

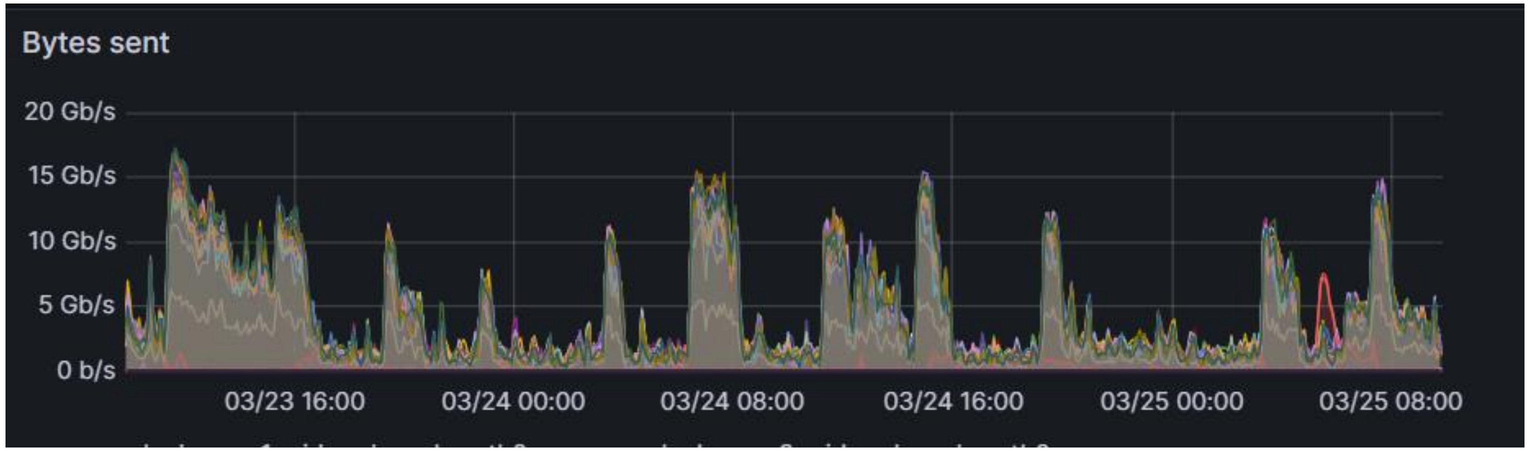
- Similar benefits to Round Robin approach, but would usually keep gateways from getting overloaded
- It’s easier to fall into the pitfalls of the existing algorithm (see appendix) and some states cannot be gotten out of without manual intervention



Phase 5 – Innovate

We decided to make our own load balancing algorithm

- Variant of weighted random load balancing
- More likely to send transfers to less loaded gateways
- A gateway will only be excluded when it goes over the allowed maximum load or is unreachable
- Working well currently



Tokens

- Token support enabled in production for ATLAS, CMS and LHCb
- Each VO uses them slightly differently….

- scope: what you're allowed to get
- filepath: what you're trying to get
- audience: who you're allowed to get it from (usually a url)

- CMS user reads - scope:/ filepath: anything in /store (the CMS root folder)
  - Restrict access to cms space only (and parse filepaths correctly)
- ATLAS - rucio audience does not include url prefixes (root:/, https:/)
  - Those audiences must be put first in the audience list to be parsed correctly
- LHCb – scope:/lhcb/user/file1, filepath:/lhcb/user
  - Allowed by the wlcg token spec[1] for creating/stating superfolders necessary for the file

[1] https://github.com/WLCG-AuthZ-WG/common-jwt-profile/blob/master/profile.md

Network tuning

Highly recommend these, significantly higher throughput achieved during DC after changes were applied

- Increase TCP buffer sizes
- Increase ring buffers
- Adaptive tx and rx on
- Enable fair queuing(fq)
- Use ecn kernel default (2)
- No CPU tuning done (to avoid clashing with previous CPU tunings done at RAL)

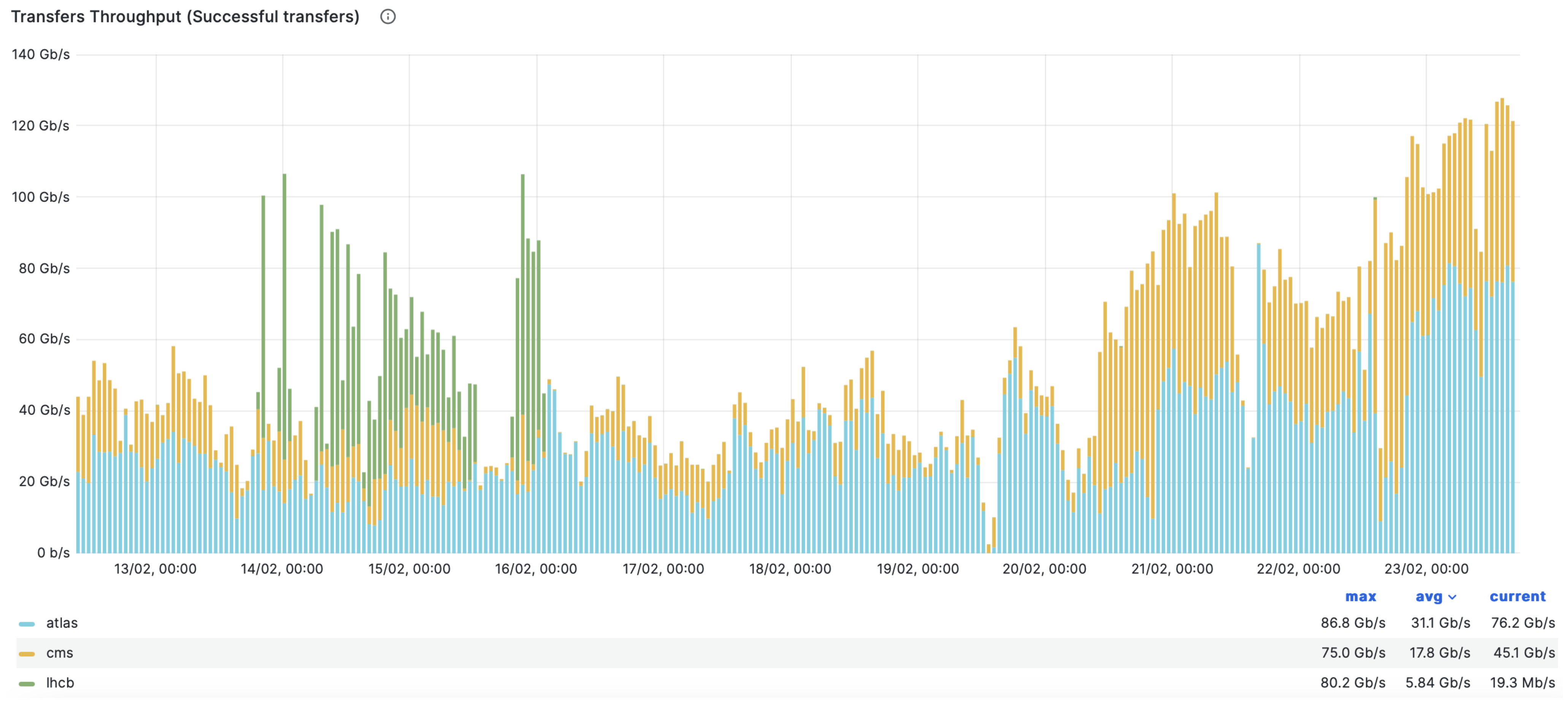
- https://fasterdata.es.net/host-tuning/linux/100g-tuning/

Future Plans

- XrootD 5.6+
  - Blocking root TPC issue against dcache sites has been resolved
- Kubernetes containerized XrootD gateway
- Additional XrootD gateways incoming
- XrootD testing framework graduate project by Mariam Demir

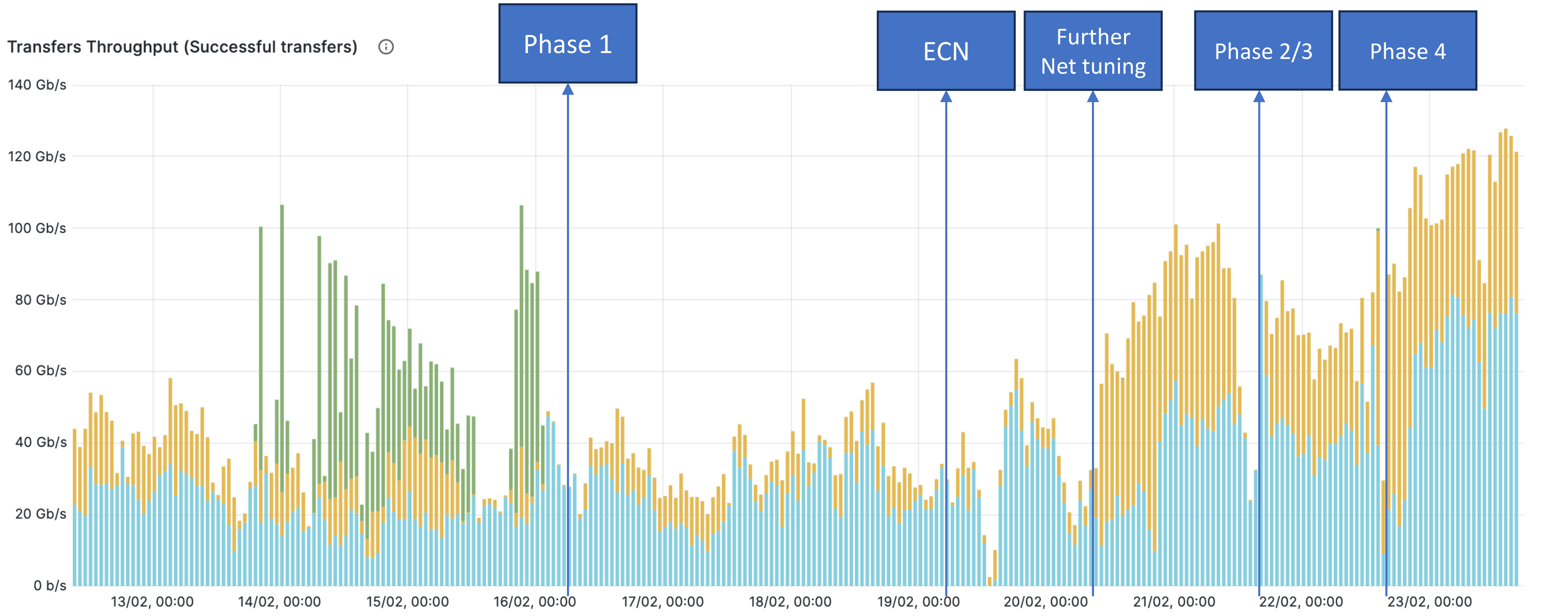
- OS upgrade (Rocky 8)
  - In testing
- 100Gbps XrootD Gateway
  - Testing ongoing
  - Jointly funded by SKA and GridPP

Transfer throughput over DC24



	max	avg	current
atlas	86.8 Gb/s	31.1 Gb/s	76.2 Gb/s
cms	75.0 Gb/s	17.8 Gb/s	45.1 Gb/s
lhcb	80.2 Gb/s	5.84 Gb/s	19.3 Mb/s

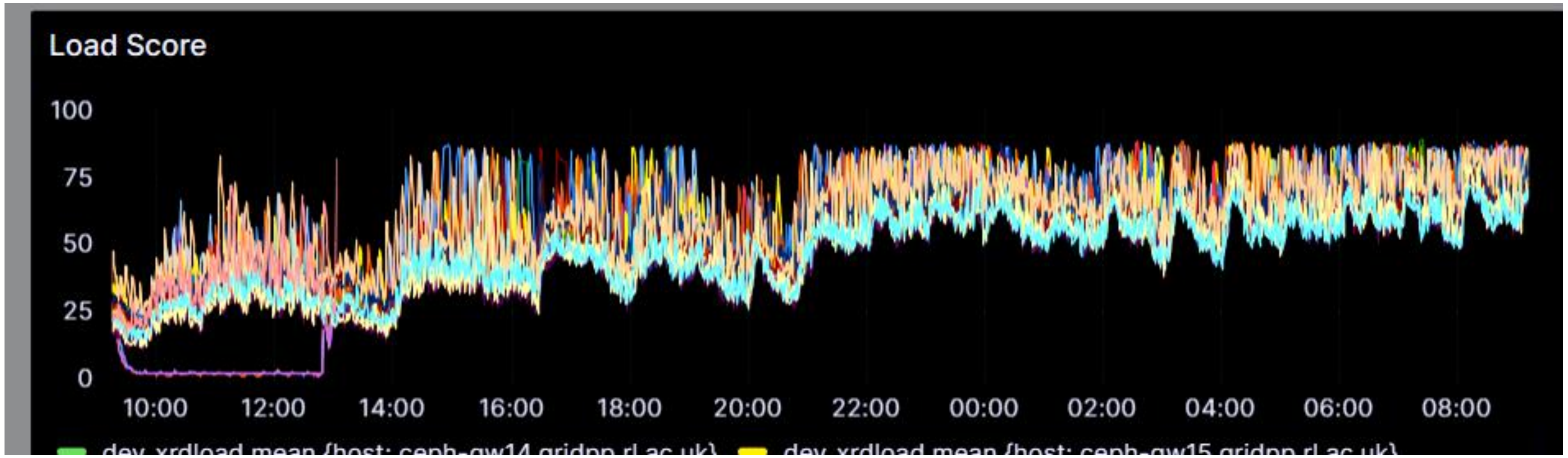
Transfer throughput over DC24



	max	avg	current
atlas	86.8 Gb/s	31.1 Gb/s	76.2 Gb/s
cms	75.0 Gb/s	17.8 Gb/s	45.1 Gb/s
lhcb	80.2 Gb/s	5.84 Gb/s	19.3 Mb/s

Thank you

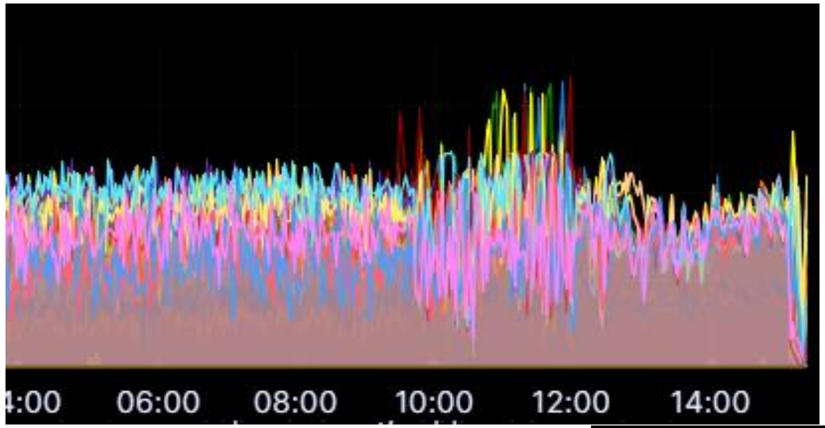
XrootD load balancing under pressure



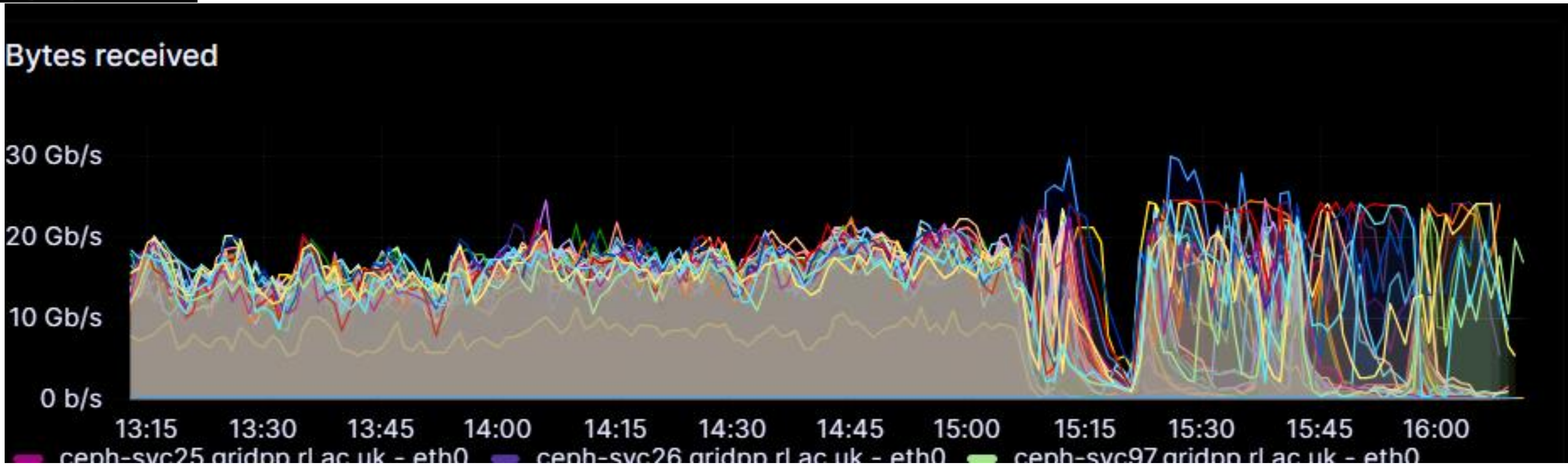
Load keeps capping out and transfers go to whichever gateways go below the maximum load first,
Pushing it back over the load limit

XrootD load based balancing vs Round Robin

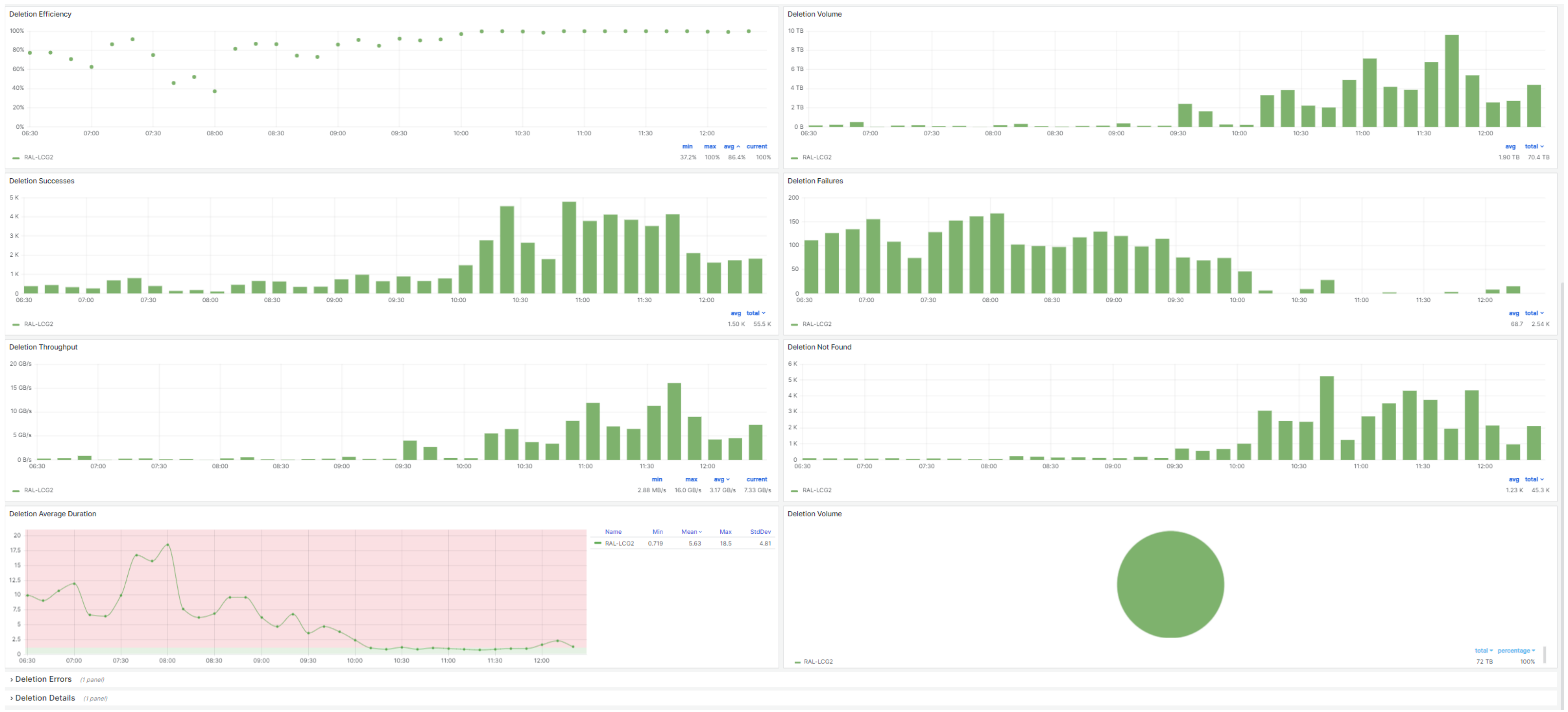
Xrootd load balancing to RR



RR to Xrootd load balancing based on nonstandard loads



Deletion efficiency default vs new algorithm



Deletion Efficiency

	min	max	avg ^	current
RAL-LCG2	37.2%	100%	86.4%	100%

Deletion Volume

	avg	total ⌄
RAL-LCG2	1.90 TB	70.4 TB

Deletion Successes

	avg	total ⌄
RAL-LCG2	1.50 K	55.5 K

Deletion Failures

	avg	total ⌄
RAL-LCG2	68.7	2.54 K

Deletion Throughput

	min	max	avg ⌄	current
RAL-LCG2	2.88 MB/s	16.0 GB/s	3.17 GB/s	7.33 GB/s

Deletion Not Found

	avg	total ⌄
RAL-LCG2	1.23 K	45.3 K

Deletion Average Duration

Name	Min	Mean ⌄	Max	StdDev
RAL-LCG2	0.719	5.63	18.5	4.81

Deletion Volume

	total ▾	percentage ▾
RAL-LCG2	72 TB	100%

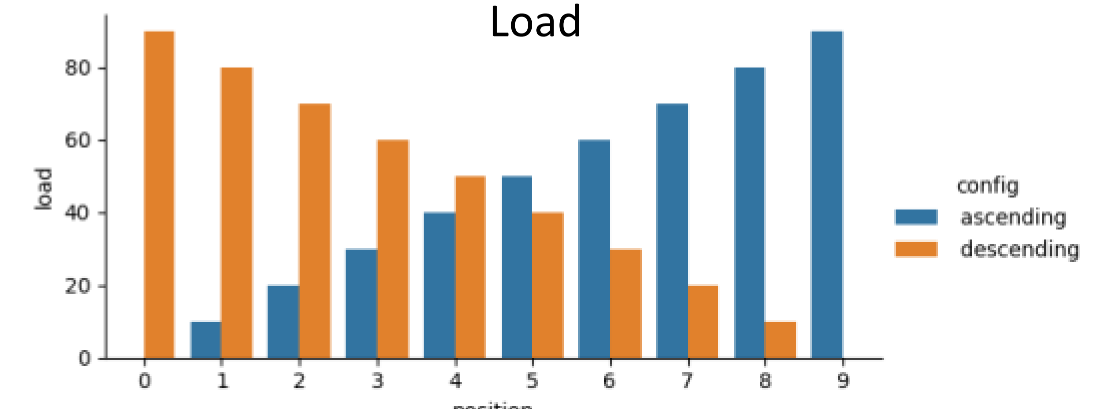
› Deletion Errors (1 panel)

› Deletion Details (1 panel)

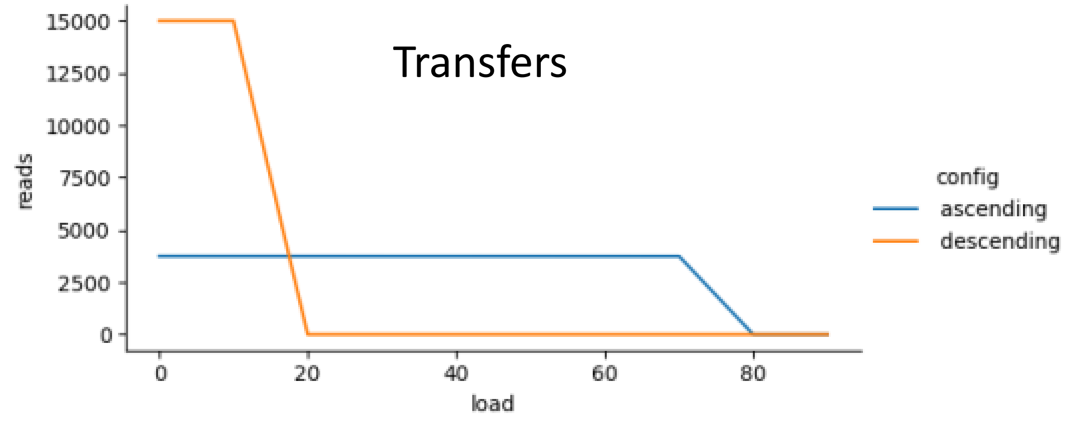
04/03/24

Problematic load patterns - Ascending/descending order of load

Load

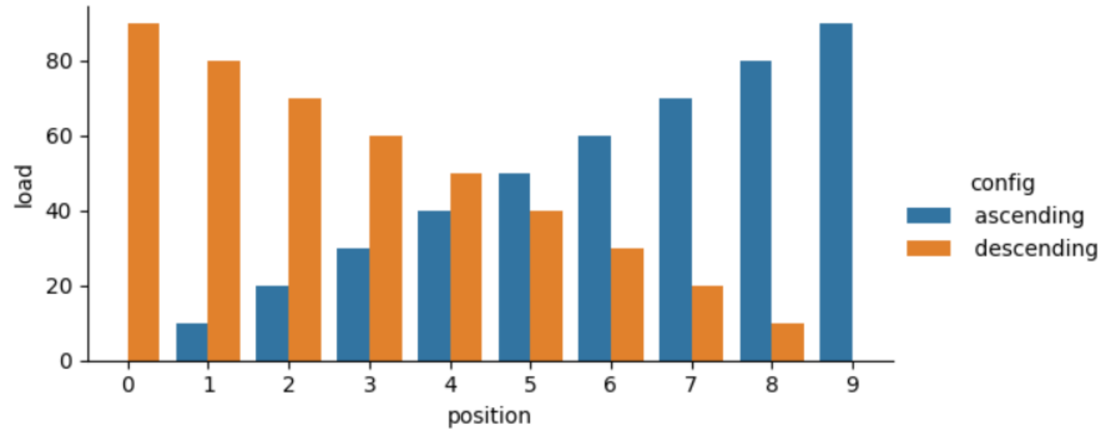


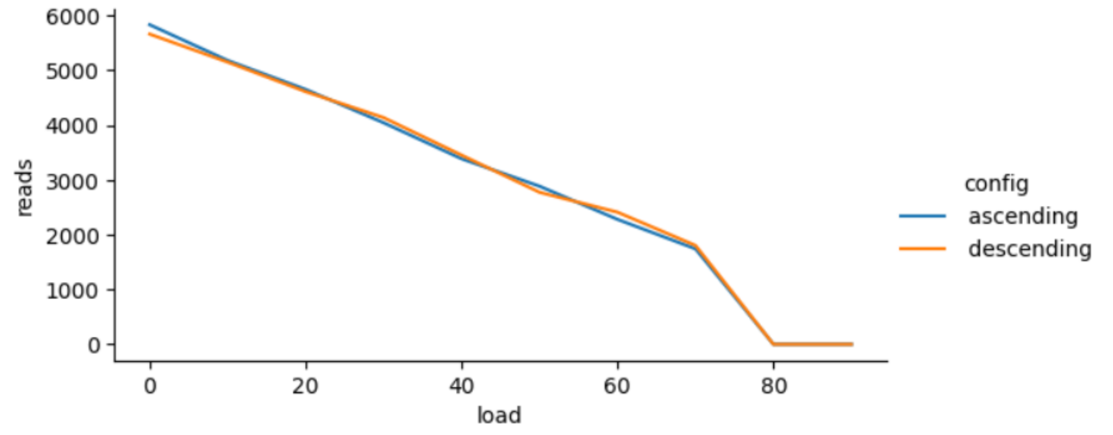
Transfers



XRootD CMSD SelByLoad Analysis, Thomas Byrne 2024

Problematic pattern- Ascending/descending order of load - new algorithm

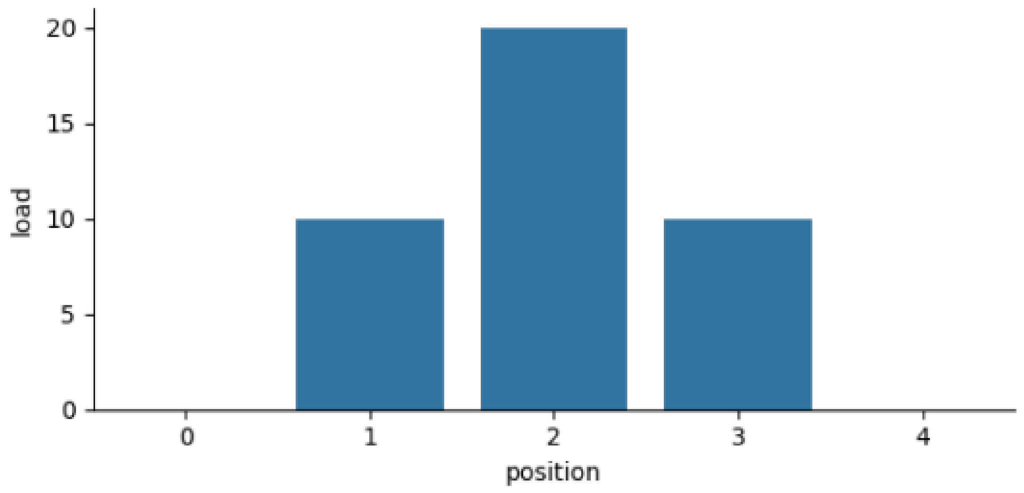




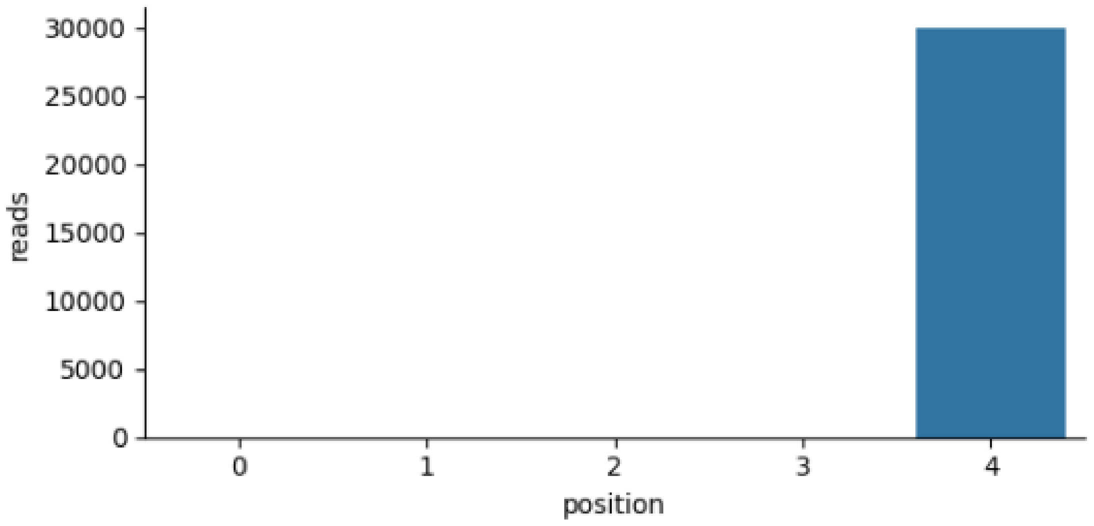
XRootD CMSD SelByLoad Analysis, Thomas Byrne 2024

Problematic load patterns- Artificial hotspotting

```
   name  load  reads
0   gw1     0      2
1   gw2    10      1
2   gw3    20      0
3   gw4    10      1
4   gw5     0  29996
```

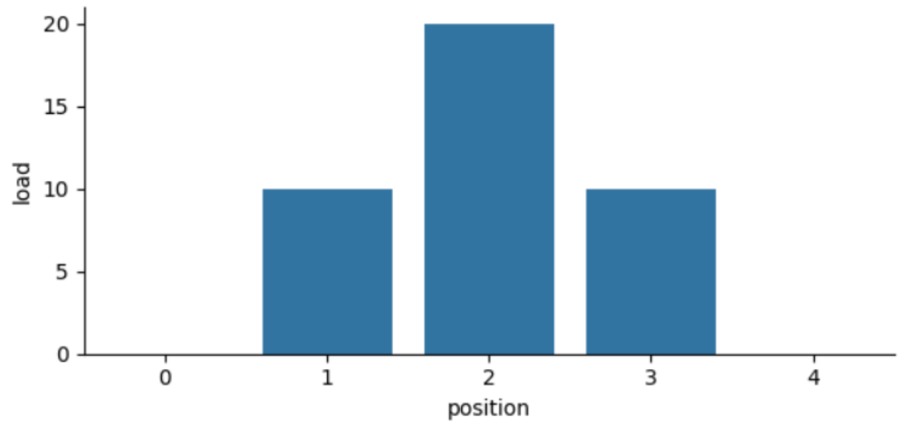


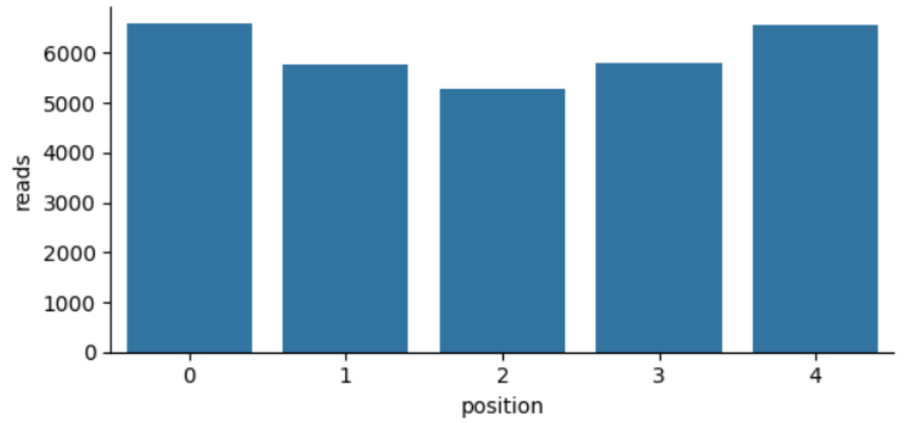
Load



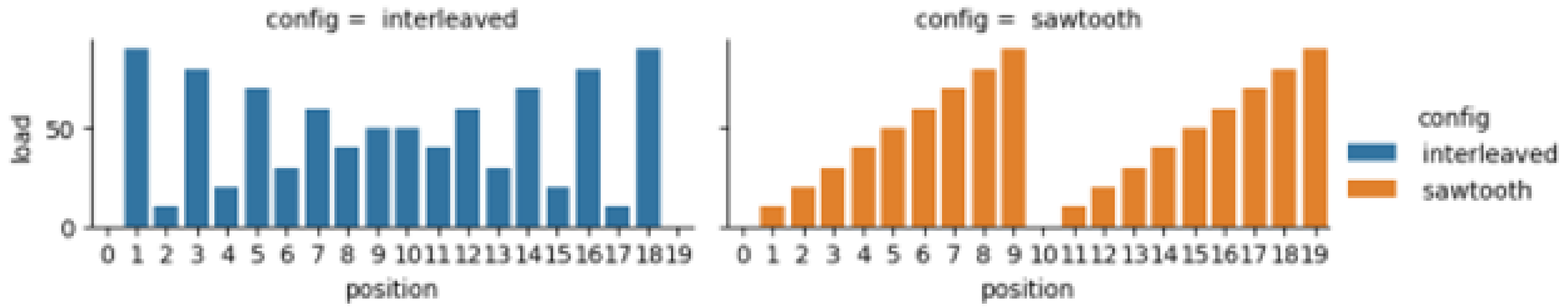
Transfers

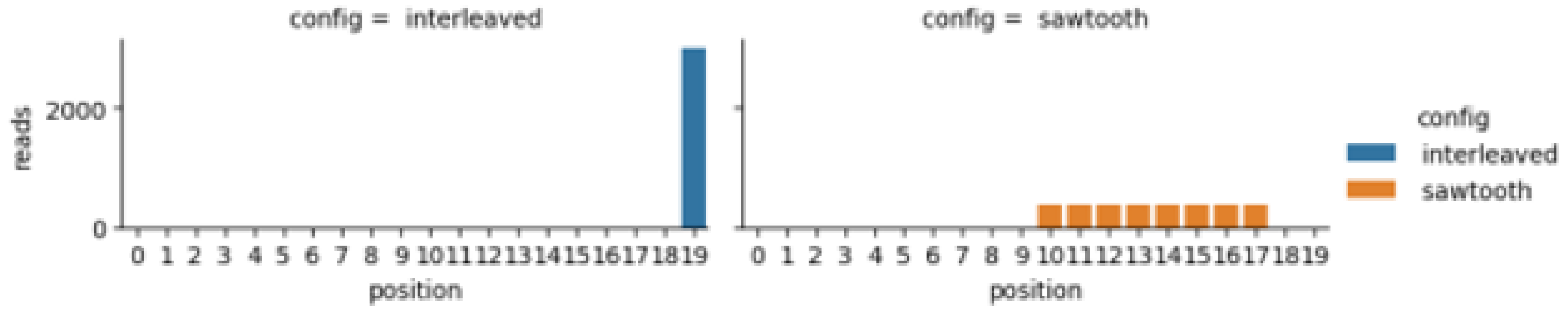
Problematic load patterns- Artificial hotspotting – new algorithm



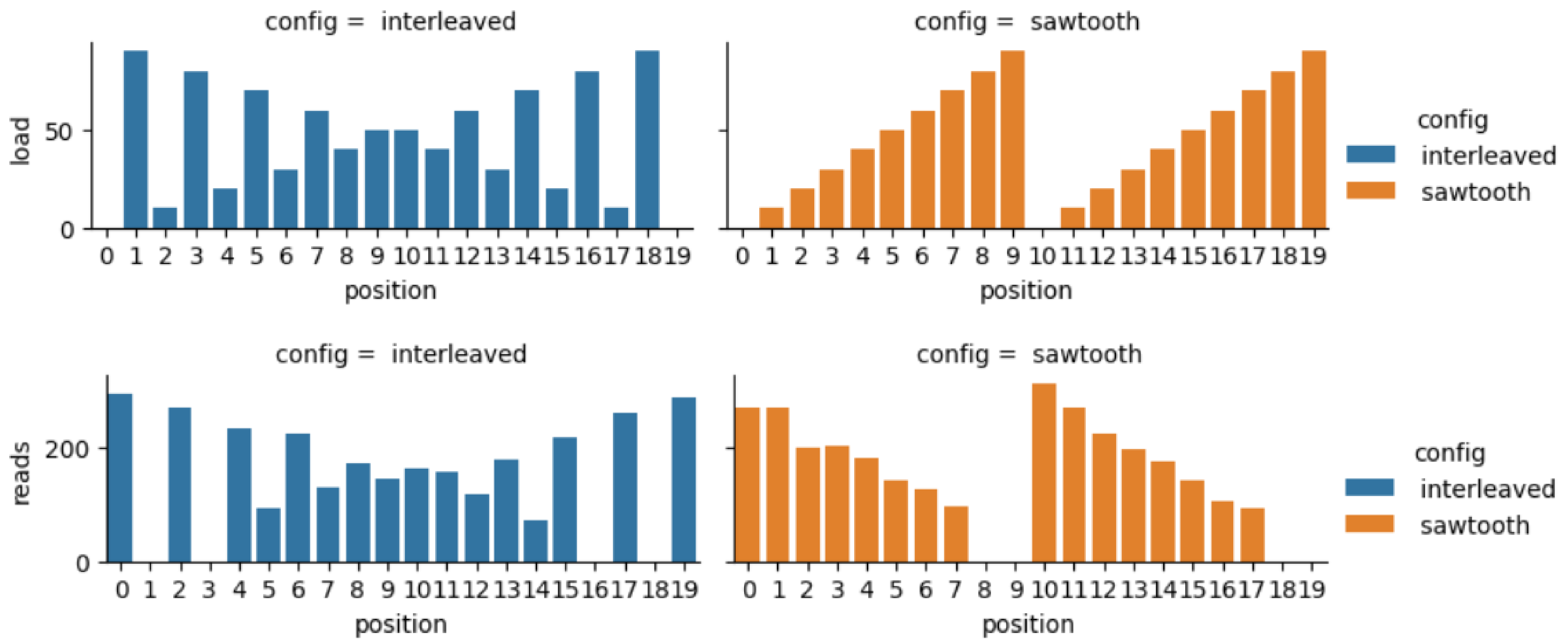


Other problematic patterns





Other problematic patterns – new algorithm



XRootD CMSD SelByLoad Analysis, Thomas Byrne 2024